11) What is classloader?

Classloader is a subsystem of JVM which is used to load class files. Whenever we run the java program, it is loaded first by the classloader. There are three built-in classloaders in Java.

1. **Bootstrap ClassLoader**: This is the first classloader which is the superclass of Extension classloader. It loads the *rt.jar* file which contains all class files of Java Standard Edition like java.lang package classes, java.net package classes, java.util package classes, java.io package classes, java.sql package classes, etc.
2. **Extension ClassLoader**: This is the child classloader of Bootstrap and parent classloader of System classloader. It loads the jar files located inside *$JAVA\_HOME/jre/lib/ext* directory.
3. **System/Application ClassLoader**: This is the child classloader of Extension classloader. It loads the class files from the classpath. By default, the classpath is set to the current directory. You can change the classpath using "-cp" or "-classpath" switch. It is also known as Application classloader.

### 30) Does constructor return any value?

**Ans:** yes, The constructor implicitly returns the current instance of the class

### 31)Is constructor inherited?

No, The constructor is not inherited

### 32) Can you make a constructor final?

No, the constructor can't be final.

### 33) Can we overload the constructors?

Yes

### 34) What do you understand by copy constructor in Java?

There is no copy constructor in java.

### 39) What is the static variable?

The static variable is used to refer to the common property of all objects

40) What is the static method?

* A static method belongs to the class rather than the object.
* There is no need to create the object to call the static methods.
* A static method can access and change the value of the static variable.

41) What are the restrictions that are applied to the Java static methods?

Two main restrictions are applied to the static methods.

* The static method can not use non-static data member or call the non-static method directly.
* this and super cannot be used in static context as they are non-static.

### 42) Why is the main method static?

Because the object is not required to call the static method. If we make the main method non-static, JVM will have to create its object first and then call main() method which will lead to the extra memory allocation.

### 43) Can we override the static methods?

No, we can't override static methods.

### 44) What is the static block?

Static block is used to initialize the static data member. It is executed before the main method, at the time of classloading.

### 46) What if the static modifier is removed from the signature of the main method?

Program compiles. However, at runtime, It throws an error "NoSuchMethodError."

### 48) Can we make constructors static?

As we know that the static context (method, block, or variable) belongs to the class, not the object. Since Constructors are invoked only when the object is created, there is no sense to make the constructors static. However, if you try to do so, the compiler will show the compiler error.

### 49) Can we make the abstract methods static in Java?

In Java, if we make the abstract methods static, It will become the part of the class, and we can directly call it which is unnecessary. Calling an undefined method is completely useless therefore it is not allowed.

### 50) Can we declare the static variables and methods in an abstract class?

Yes, we can declare static variables and methods in an abstract method. As we know that there is no requirement to make the object to access the static context, therefore, we can access the static context declared inside the abstract class by using the name of the abstract class.

### 51) What is ****this**** keyword in java?

The **this** keyword is a reference variable that refers to the current object.

### 53) Can we assign the reference to ****this**** variable?

No, this cannot be assigned to any value because it always points to the current class object and this is the final reference in Java. However, if we try to do so, the compiler error will be shown

### 54) Can ****this**** keyword be used to refer static members?

Yes, It is possible to use this keyword to refer static members because this is just a reference variable which refers to the current class object. However, as we know that, it is unnecessary to access static variables through objects, therefore, it is not the best practice to use this to refer static members.

56) What are the advantages of passing this into a method instead of the current class object itself?

As we know, that this refers to the current class object, therefore, it must be similar to the current class object. However, there can be two main advantages of passing this into a method instead of the current class object.

* this is a final variable. Therefore, this cannot be assigned to any new value whereas the current class object might not be final and can be changed.
* this can be used in the synchronized block.

### 61) What is aggregation?

Aggregation can be defined as the relationship between two classes where the aggregate class contains a reference to the class it owns. Aggregation is best described as a **has-a** relationship. For example, The aggregate class Employee having various fields such as age, name, and salary also contains an object of Address class having various fields such as Address-Line 1, City, State, and pin-code. In other words, we can say that Employee (class) has an object of Address class.

### 62) What is composition?

Holding the reference of a class within some other class is known as composition. When an object contains the other object, if the contained object cannot exist without the existence of container object, then it is called composition. In other words, we can say that composition is the particular case of aggregation which represents a stronger relationship between two objects. Example: A class contains students. A student cannot exist without a class. There exists composition between class and students.

### 63) What is the difference between aggregation and composition?

Aggregation represents the weak relationship whereas composition represents the strong relationship. For example, the bike has an indicator (aggregation), but the bike has an engine (composition).

68) What are the differences between this and super keyword?

There are the following differences between this and super keyword.

* The super keyword always points to the parent class contexts whereas this keyword always points to the current class context.
* The super keyword is primarily used for initializing the base class variables within the derived class constructor whereas this keyword primarily used to differentiate between local and instance variables when passed in the class constructor.
* The super and this must be the first statement inside constructor otherwise the compiler will throw an error.

### 76) What is method overloading with type promotion?

By Type promotion is method overloading, we mean that one data type can be promoted to another implicitly if no exact matching is found.



# **Virtual Function in Java**

A virtual function or virtual method in an OOP language is a function or method used to override the behavior of the function in an inherited class with the same signature to achieve the polymorphism.

By default, all the instance methods in Java are considered as the Virtual function except final, static, and private methods as these methods can be used to achieve polymorphism.

93) What is the final blank variable?

A final variable, not initialized at the time of declaration, is known as the final blank variable. We can't initialize the final blank variable directly. Instead, we have to initialize it by using the class constructor. It is useful in the case when the user has some data which must not be changed by others, for example, PAN Number. Consider the following example:

1. **class** Student{
2. **int** id;
3. String name;
4. **final** String PAN\_CARD\_NUMBER;
5. ...
6. }

### 98) Can we declare a constructor as final?

The constructor can never be declared as final because it is never inherited.

### 99) Can we declare an interface as final?

No, we cannot declare an interface as final because the interface must be implemented by some class to provide its definition.

### 100) What is the difference between the final method and abstract method?

The main difference between the final method and abstract method is that the abstract method cannot be final as we need to override them in the subclass to give its definition.

### 121) How to make a read-only class in Java?

A class can be made read-only by making all of the fields private. The read-only class will have only getter methods which return the private property of the class to the main method.

### 122) How to make a write-only class in Java?

A class can be made write-only by making all of the fields private. The write-only class will have only setter methods which set the value passed from the main method to the private fields.

### 101) What is the difference between compile-time polymorphism and runtime polymorphism?

|  |  |  |
| --- | --- | --- |
| **SN** | **compile-time polymorphism** | **Runtime polymorphism** |
| 1 | In compile-time polymorphism, call to a method is resolved at compile-time. | In runtime polymorphism, call to an overridden method is resolved at runtime. |
| 2 | It is also known as static binding, early binding, or overloading. | It is also known as dynamic binding, late binding, overriding, or dynamic method dispatch. |
| 3 | Overloading is a way to achieve compile-time polymorphism in which, we can define multiple methods or constructors with different signatures. | Overriding is a way to achieve runtime polymorphism in which, we can redefine some particular method or variable in the derived class. By using overriding, we can give some specific implementation to the base class properties in the derived class. |
| 4 | It provides fast execution because the type of an object is determined at compile-time. | It provides slower execution as compare to compile-time because the type of an object is determined at run-time. |
| 5 | Compile-time polymorphism provides less flexibility because all the things are resolved at compile-time. | Run-time polymorphism provides more flexibility because all the things are resolved at runtime. |

103) Can you achieve Runtime Polymorphism by data members?

No, because method overriding is used to achieve runtime polymorphism and data members cannot be overridden. We can override the member functions but not the data members. Consider the example given below.

1. **class** Bike{
2. **int** speedlimit=90;
3. }
4. **class** Honda3 **extends** Bike{
5. **int** speedlimit=150;
6. **public** **static** **void** main(String args[]){
7. Bike obj=**new** Honda3();
8. System.out.println(obj.speedlimit);//90
9. }

### 104) What is the difference between static binding and dynamic binding?

In case of the static binding, the type of the object is determined at compile-time whereas, in the dynamic binding, the type of the object is determined at runtime.

**Static Binding**

1. **class** Dog{
2. **private** **void** eat(){System.out.println("dog is eating...");}
4. **public** **static** **void** main(String args[]){
5. Dog d1=**new** Dog();
6. d1.eat();
7. }
8. }

**Dynamic Binding**

1. **class** Animal{
2. **void** eat(){System.out.println("animal is eating...");}
3. }
5. **class** Dog **extends** Animal{
6. **void** eat(){System.out.println("dog is eating...");}
8. **public** **static** **void** main(String args[]){
9. Animal a=**new** Dog();
10. a.eat();
11. }
12. }

106) What is Java instanceOf operator?

The instanceof in Java is also known as type comparison operator because it compares the instance with type. It returns either true or false. If we apply the instanceof operator with any variable that has a null value, it returns false.